13019 – Diseño de bases de datos

Capítulo 2 – Modelado orientado a objetos

Wladimiro Díaz

Wladimiro.Diaz@uv.es

Universitat de València

El modelo orientado a objetos

- La década de los 90: la era de la programación orientada a objetos.
  - **Necesidad de este paradigma**: los usuarios demandan programas y entornos de trabajo simples y fáciles de usar.
  - **Implicaciones**: mayor número de líneas de código que es necesario organizar, gestionar y mantener.
- Proporciona mejores herramientas para:
  - Obtener un modelo del mundo real cercano a la perspectiva del usuario.
  - Interaccionar fácilmente con un entorno de computación, empleando metáforas familiares.
  - Facilitar la modificación y la extensión de los componentes sin codificar de nuevo desde cero.

El modelo orientado a objetos

Las técnicas de orientación a objetos pretenden satisfacer tanto las necesidades de los usuarios finales como las de los desarrolladores de software mediante una cierta capacidad de modelar el mundo real.

El modelo orientado a objetos

- La programación *tradicional* está orientada a los procedimientos.
- En la programación orientada a objetos las entidades centrales son los datos (*objetos*).
  - Los objetos se comunican entre sí mediante el uso de mensajes y el conjunto de objetos que responden a los mismos mensajes se implementan mediante *clases*.
  - La *clase* describe e implementa todos los métodos que capturan el comportamiento de sus instancias.
  - La implementación está totalmente oculta (*encapsulada*) dentro de la clase, de modo que puede ser extendida y modificada sin afectar al usuario.
  - Una clase es como un módulo. Sin embargo, también es posible extender y especializar una clase (mecanismo de herencia).

Ambito del modelo orientado a objetos

Los principios de la tecnología orientada o objetos se aplican a todos los aspectos del proceso de desarrollo del *software*:

- Metodología.
- Herramientas de diseño y análisis.
- Interfaces de usuario.
- Lenguajes de programación.
- Bases de datos.
- Sistemas operativos.
- . . .

Terminología común y principios básicos

- **Modularización** — Módulos fáciles de manejar y que comprenden las estructuras de datos y las operaciones permisibles.
- **Encapsulado** — Distingue entre la interfase a un objeto (*qué* es lo que hace), de la implementación (*cómo* lo hace).
- **Tipos de datos abstractos** — Agrupa todos los objetos que tienen la misma interfase y los trata como si fueran del mismo tipo.
- **Herencia** — Reutilización, ya que permite definir nuevos tipos en funciones de otros tipos. El nuevo tipo hereda las estructuras de datos y los métodos del tipo precedente.

Terminología común y principios básicos

- **Mensajes** — Un objeto lleva a cabo sus acciones cuando recibe un mensaje concreto, codificado de una forma simple, estándar e independiente de cómo o dónde está implementado el objeto.
- **Polimorfismo** — Diferentes objetos responden al mismo mensaje. El sistema determina en tiempo de ejecución qué código invocar dependiendo del tipo de objeto (técnicas de *Overloading* y *Dynamic binding*).

Conceptos de objeto y clase

Un objeto es un concepto, abstracción o cosa que tiene un cierto significado para una aplicación

- Se presentan como nombres propios o referencias específicas en la descripción o discusión de un problema:
- Algunos objetos tienen una entidad real (por ejemplo la empresa Seat).
- Otros son entidades conceptuales (por ejemplo, la fórmula para resolver una ecuación de segundo grado).
- Existen objetos que se introducen por razones de implementación y carecen de equivalencia en la realidad física (por ejemplo, un árbol binario).

Conceptos de objeto y clase

- Cada objeto tiene existencia propia y puede ser identificado. Se ha definido la *identidad* como:

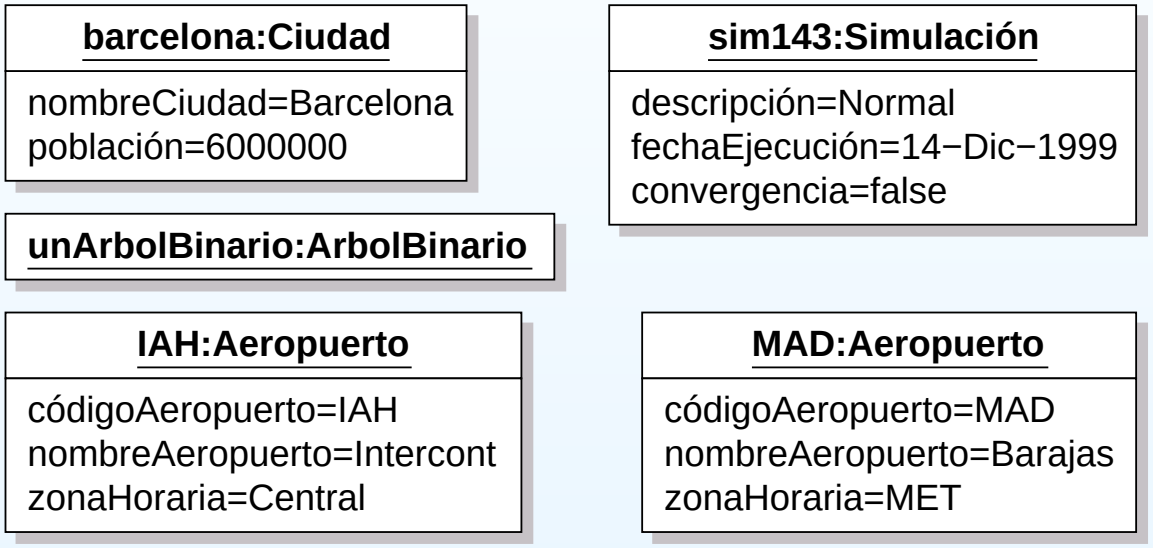
  > “aquella propiedad de un objeto que lo distingue del resto de objetos”

- En la etapa de análisis de una aplicación es posible dar por supuesto que los objetos tienen identidad.
- En el diseño, es necesario adoptar una metodología para implementar dicha identidad (direcciones de memoria, códigos numéricos o una combinación de los valores de ciertos atributos).

> Esta noción de identidad es una parte integral y muy importante de la tecnología orientada a objetos

Objetos

- Los objetos y sus relaciones se describen mediante diagramas de instancias:



- *unArbolBinario* pertenece a la clase **ArbolBinario** y no se han especificado los valores de los atributos.
- El objeto *IAH* pertenece a la clase **Aeropuerto** y tiene los valores *IAH*, *Intercont* y *Central*.

Objetos

- Cuando se construye un modelo es necesario decidir qué objetos mostrar y cuales ignorar.
- Un objeto representa sólo los aspectos relevantes de un problema.
- No resulta útil modelar detalles irrelevantes o extraños, ya que el ámbito de un objeto depende sólo de qué necesita la aplicación.

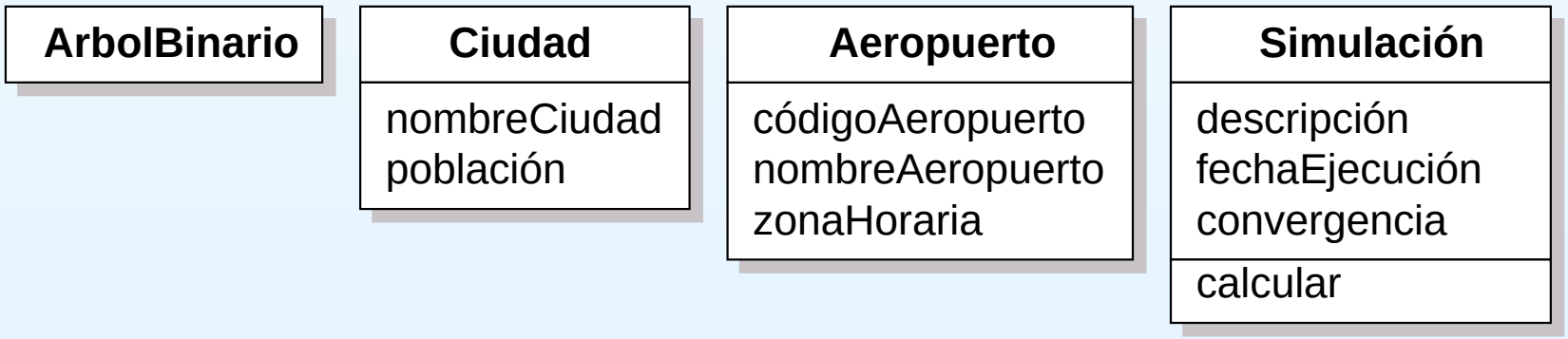
Clases

- Un objeto es una *instancia* (u ocurrencia) de una clase.
- Una *Clase* es la descripción de un grupo de objetos con:
  - Propiedades similares (atributos del objeto).
  - Comportamiento (operaciones y diagramas de estado) y semántica común.
  - Y que establecen el mismo tipo de relaciones con otros objetos.

Las clases proporcionan un mecanismo para compartir la estructura entre objetos similares

Clases

- Algunas clases tienen una contrapartida real (persona y empresa por ejemplo).
- Otras son entidades conceptuales (ecuación algebraica).
- Además, existen clases que son sólo artefactos de una implementación específica (por ejemplo, árbol binario).
- Las clases y sus relaciones se describen mediante diagramas de clases:



Clases

- No se especifican ni los atributos ni las operaciones de la clase **ArbolBinario**.
- La clase **Ciudad** tiene los atributos *nombreCiudad* y *población*.
- La clase **Aeropuerto** cuenta con los atributos *códigoAeropuerto*, *nombreAeropuerto* y *zonaHoraria*.
- No se muestran las operaciones de **Ciudad** y **Aeropuerto**. Esto no significa que ambas clases carezcan de operaciones, sino sólo que se ha decidido no mostrarlas.
- La clase **Simulación** tiene los atributos *descripción*, *fechaEjecución* y *convergencia* así como la operación *calcular*.

Diagramas de clases y objetos

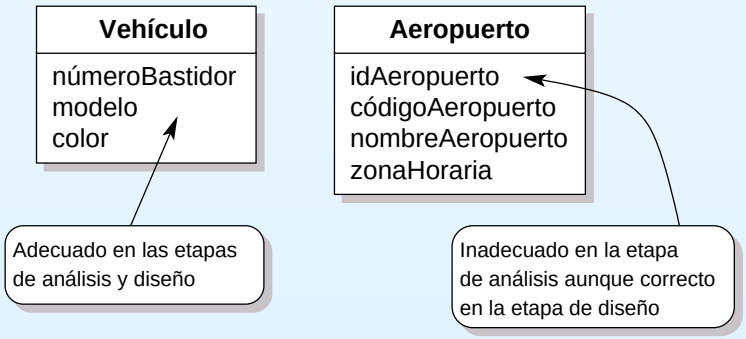
- Tanto los diagramas de clases como los de instancias constituyen una expresión del modelo orientado a objetos.
- Del mismo modo que una clase describe un conjunto de objetos, un diagrama de clases describe un conjunto de diagramas de instancias.
- Un diagrama de clases permite representar de forma abstracta el conjunto de diagramas de instancias, documentando la estructura de los datos.

Valores y atributos de objeto

- Un *valor* es un trozo de información (dato).
- Un *atributo de objeto* es una propiedad de una clase a la que se le asigna un nombre y que contiene un valor para cada objeto de la clase.
- Los modelos orientados a objetos se construyen sobre estructuras: clases y relaciones.
- Los atributos tienen una importancia menor y se utilizan para elaborar las clases y las relaciones.
- Es importante no confundir valores y objetos: Los objetos tienen identidad mientras que los valores no.

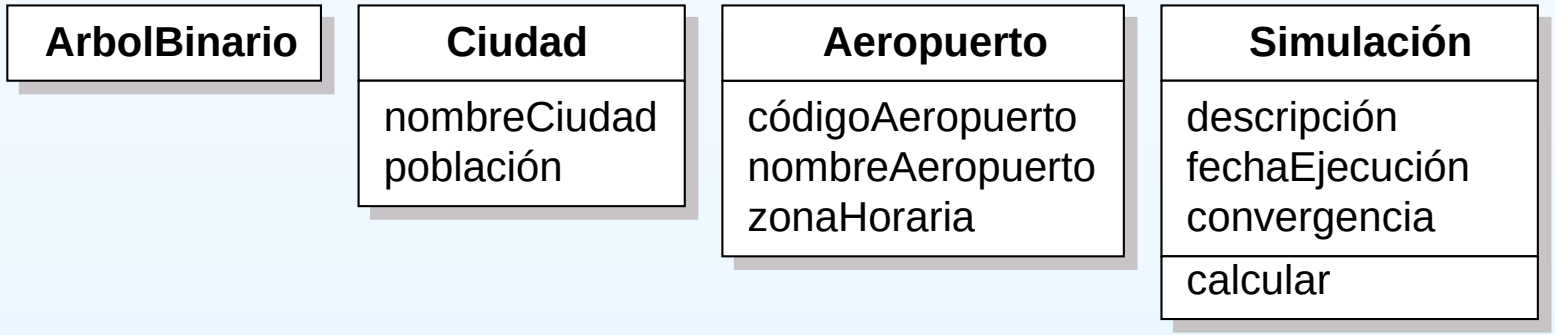
Valores y atributos de objeto

- En el análisis de un modelo: no es necesario añadir un atributo que actúe como identificador interno de las instancias de una clase.
  - Los identificadores del objeto (OIDs) son implícitos a la metodología orientada a objetos.
  - Sólo utilizamos aquellos atributos que tienen sentido para la aplicación.
- Durante la etapa de diseño puede ser necesario utilizar identificadores internos, por ejemplo cuando se utiliza un modelo relacional.



Operaciones y métodos

- Una *operación* es una función o procedimiento que se puede aplicar por un objeto o sobre un objeto.
- Cada operación actúa sobre un objeto que es su argumento implícito.



- La clase **Simulación** contiene la operación *calcular* cuyo argumento implícito es un objeto de la clase **Simulación**.

Operaciones y métodos

- Al nombre de la operación se pueden añadir detalles opcionales tales como la lista de argumentos o el tipo del resultado.
- Algunas operaciones son *polimórficas*, esto es, aplicables a muchas clases. La implementación de una operación para una clase concreta se denomina *método*.

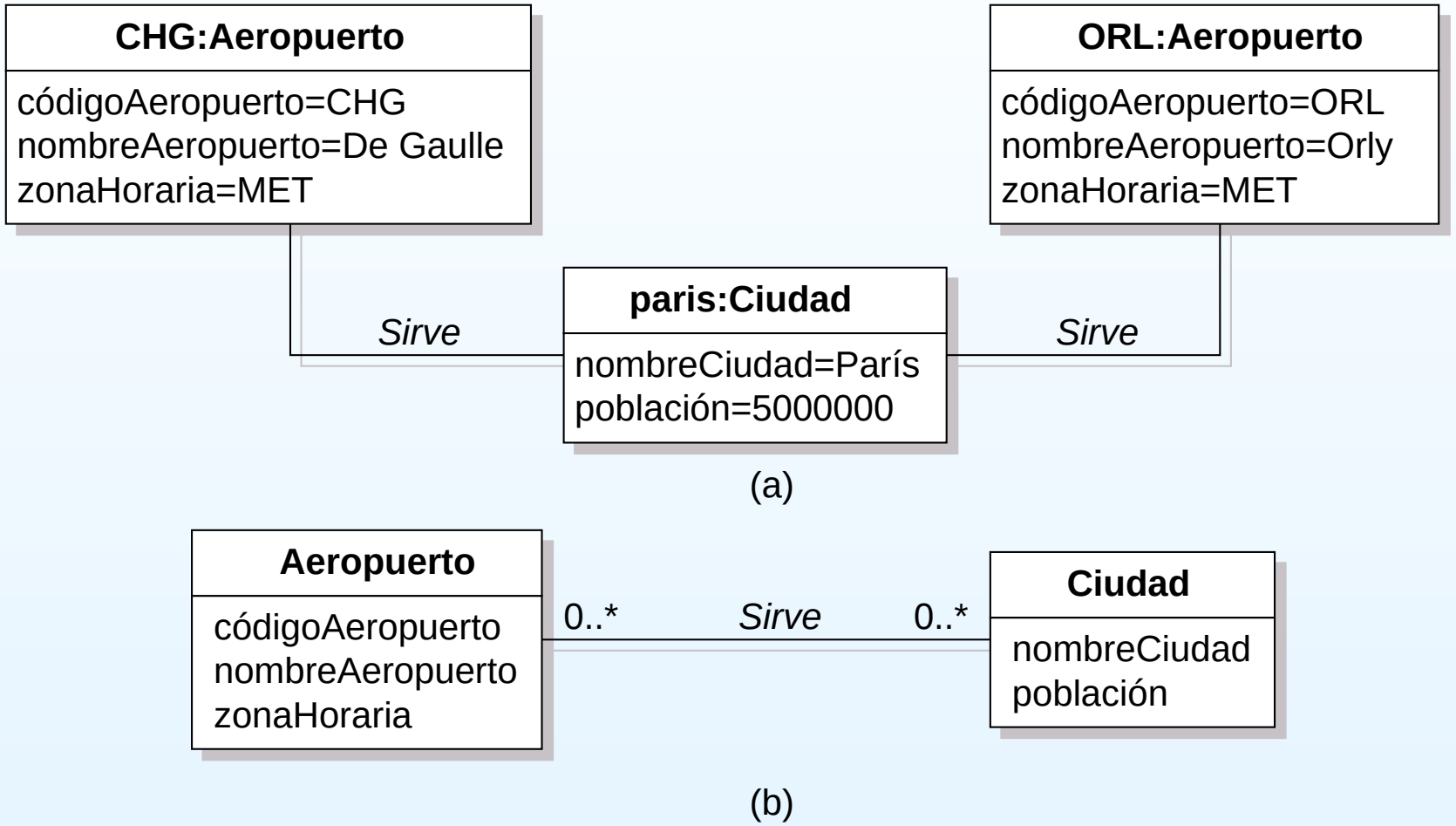
Conceptos de enlace y asociación

- Un *enlace* (*link*) es una conexión física o conceptual entre objetos. Muchos enlaces interconectan dos objetos, pero es posible la existencia de enlaces entre tres o más objetos.
- Una *asociación* es la descripción de un grupo de enlaces con una estructura y semántica común.

De acuerdo con esto:

- Un enlace es una instancia de una asociación. Los enlaces de una asociación relacionan objetos relacionan objetos de las mismas clases y tienen propiedades similares (atributos del enlace).
- Una asociación describe un conjunto de enlaces potenciales del mismo modo que una clase describe un conjunto potencial de objetos.

Conceptos de enlace y asociación

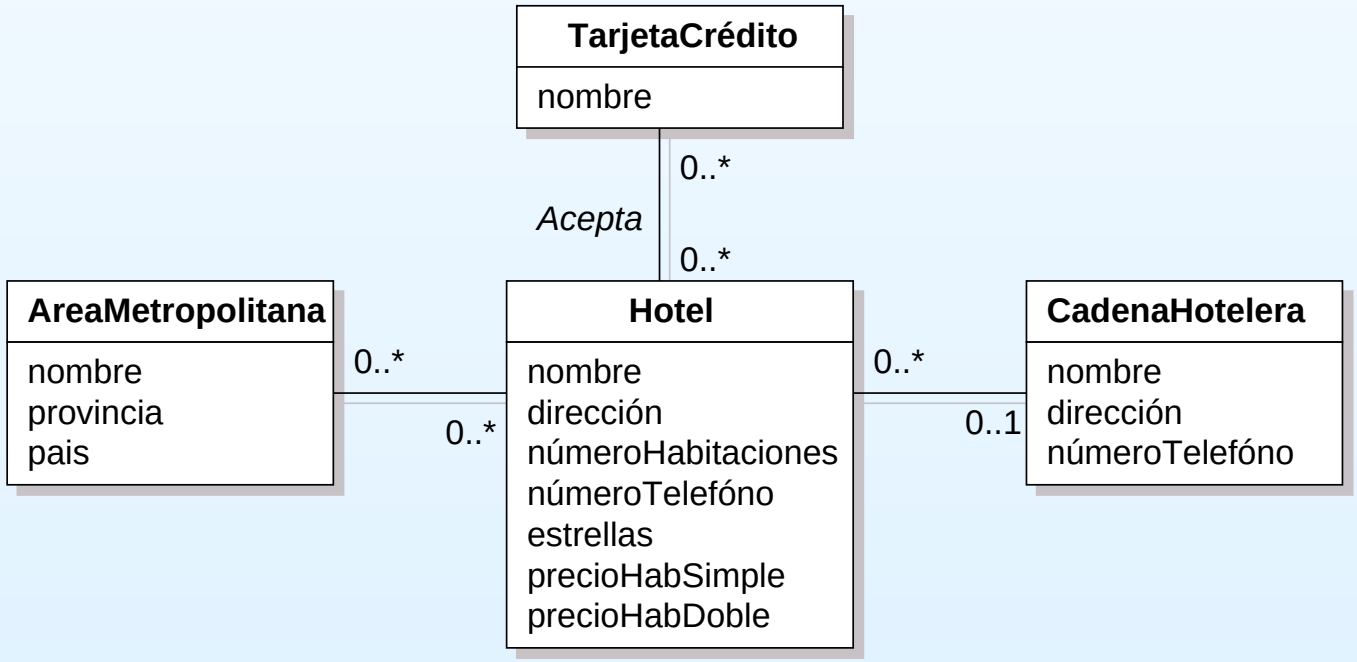


Conceptos de enlace y asociación

- Es importante no confundir un enlace con un objeto.
  - Los objetos tienen una identidad inherente y existen por derecho propio.
  - La identidad de un enlace deriva de los objetos que relaciona y no puede existir de forma aislada.
- En la etapa de análisis, todas las relaciones entre objetos se modelan mediante enlaces y los grupos de enlaces similares se reconocen como asociaciones.
- En la etapa de diseño, las asociaciones se pueden implementar mediante punteros, cláves foráneas o mediante cualquier otro método.

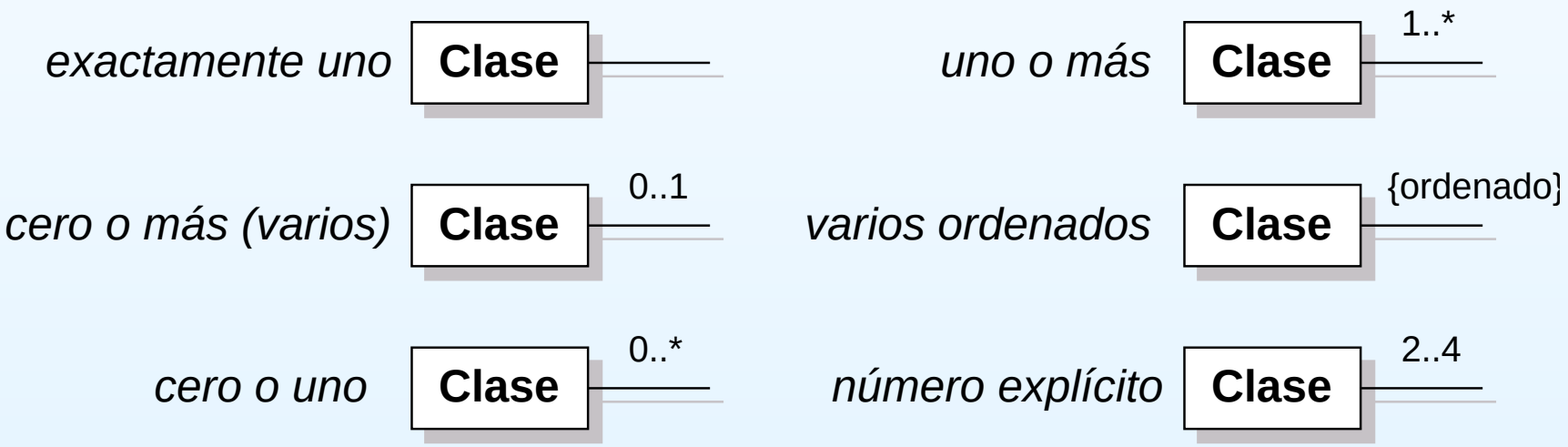
Conceptos de enlace y asociación

- Las asociaciones constituyen un aspecto importante y distintivo del modelado OMT (y UML).
  - Muchas metodologías de desarrollo y lenguajes orientados a objetos han pasado por alto las asociaciones.
  - Una asociación no es lo mismo que un puntero.

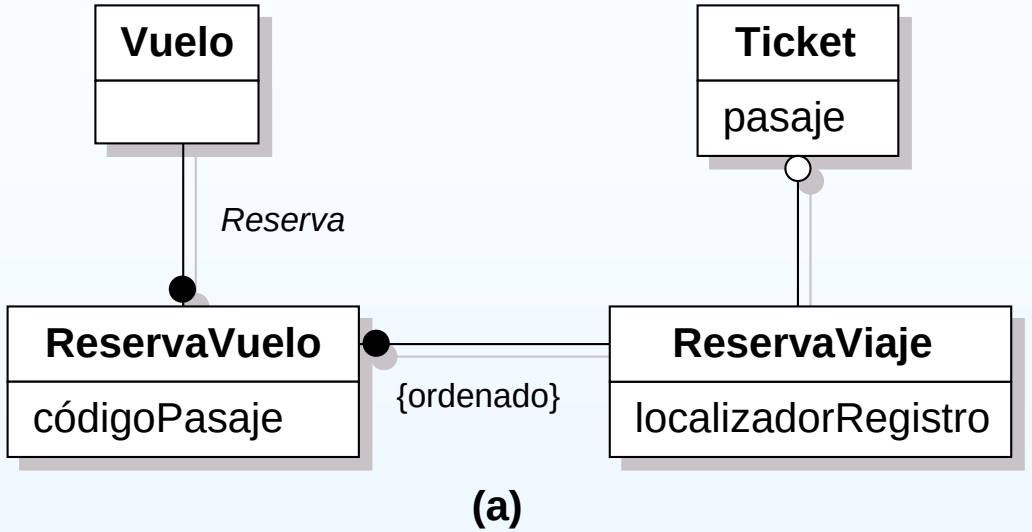


Multiplicidad

La *multiplicidad* especifica el número de instancias de una clase que se pueden relacionar con una instancia de la clase relacionada



Multiplicidad

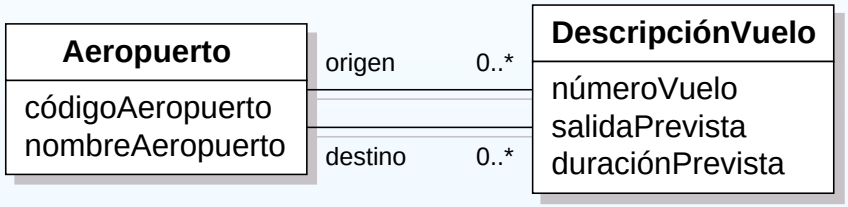


Multiplicidad

- Un vuelo puede ser reservado por múltiples reservas de vuelo.
- El código de pasaje de un vuelo reservado determina el precio del pasaje que cobrará la línea aérea.
- Una reserva de viaje consiste en múltiples reservas de vuelo que están ordenadas a medida que un pasajero viaja de una ciudad hasta la siguiente.
- La línea aérea utiliza un localizador de registro para identificar una reserva de viaje en su sistema de información.

Roles

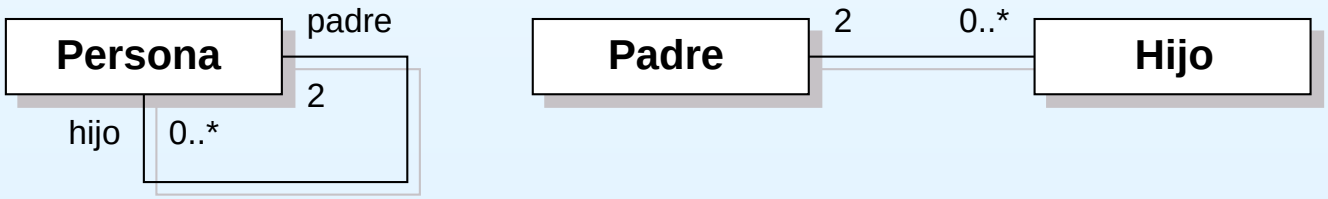
- Un *rol* es uno de los extremos de una asociación al que se le puede asignar un nombre.



- Los nombres de *rol* son especialmente útiles en el recorrido de las asociaciones debido a que se pueden tratar como pseudo-atributos:
  - *unaDescripciónVuelo.origen* hace referencia al aeropuerto de origen del vuelo.
  - *unaDescripciónVuelo.destino* hace referencia al aeropuerto en el que finaliza el vuelo.

Roles

- Ya que el *rol* es un pseudo-atributo, el nombre de *rol* no puede entrar en contradicción con cualquier otro atributo o *rol* de la clase que lo origina.
- Los nombres de *rol* son opcionales si el modelo no es ambigüo. La ambigüedad puede aparecer cuando:
  - Se dan múltiples asociaciones entre las mismas clase (como hemos visto en la figura anterior).
  - Cuando la asociación tiene lugar entre objetos de la misma clase (*asociación reflexiva*, como se muestra en la figura adjunta.

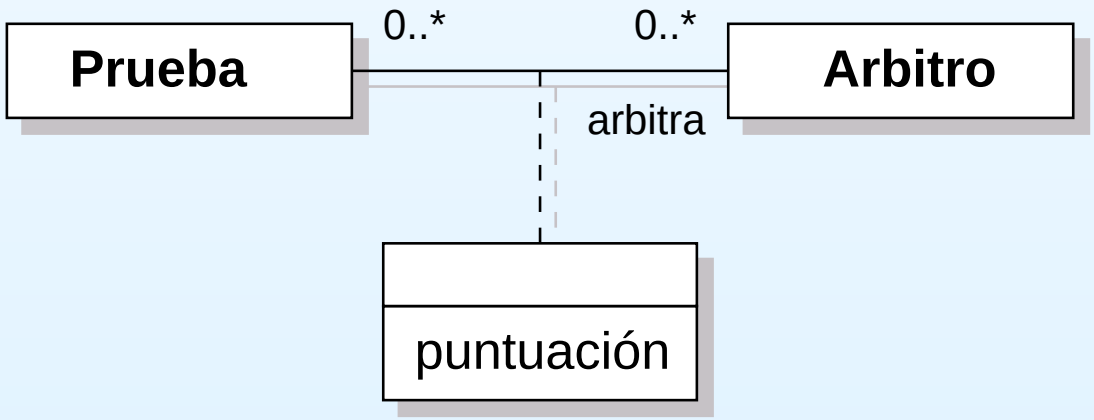


Modelo correcto

Modelo incorrecto

Atributos de enlace

- El atributo de un objeto es una propiedad de la clase con un nombre que describe un valor contenido por cada objeto de la clase.
- De modo similar, un *atributo de enlace* es una propiedad de una asociación con nombre que describe un valor contenido por cada enlace de la asociacón.
- Un atributo de enlace es una propiedad de la asociación no se puede adscribir a ninguna de las clases asociadas.



Atributos de enlace

- un árbitro asigna una puntuación a los esfuerzos realizados por un competidor (atleta) de una prueba de decatlon.
- La *puntuación* es el atributo de la asociación entre las clases **Prueba** y **Arbitro**.
- Las asociaciones muchos-a-muchos proporcionan el argumento más convincente y la razón fundamental de los atributos de enlace.
  - Un atributo de este tipo es sin duda alguna del enlace y no puede asignarse a ninguno de los dos objetos relacionados.

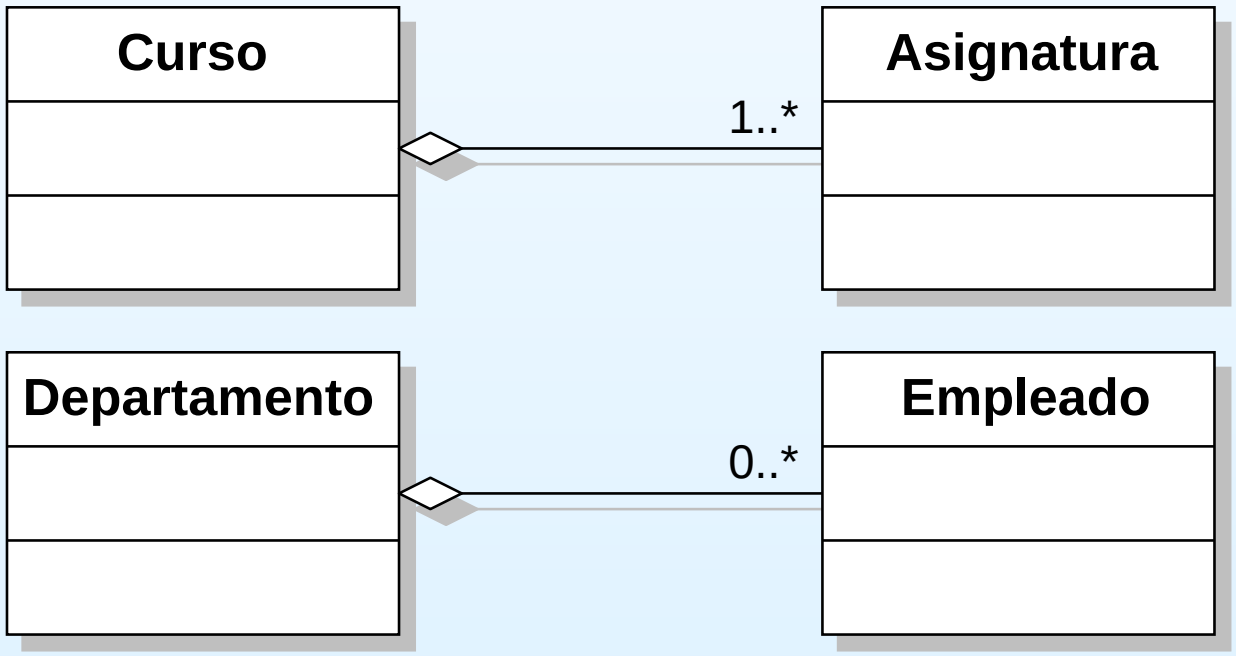
La puntuación depende tanto del atleta como del árbitro

Agregación, composición y propiedad

- La **agregación** es una relación del tipo parte–todo entre dos clases.
- UML permite definir dos tipos de agregación:
  - **Agregación simple**: también denominada *agregación compartida*. Permite modelar una relación parte–todo en la cual un objeto es propietario de otro objeto pero no en exclusividad. El objeto poseído puede ser a su vez poseído por otros objetos.
  - **Composición**. También denominada en ocasiones *agregación fuerte* o *agregación compuesta*. Permite modelar una relación parte–todo en la que un objeto es propietario en exclusiva del otro objeto.

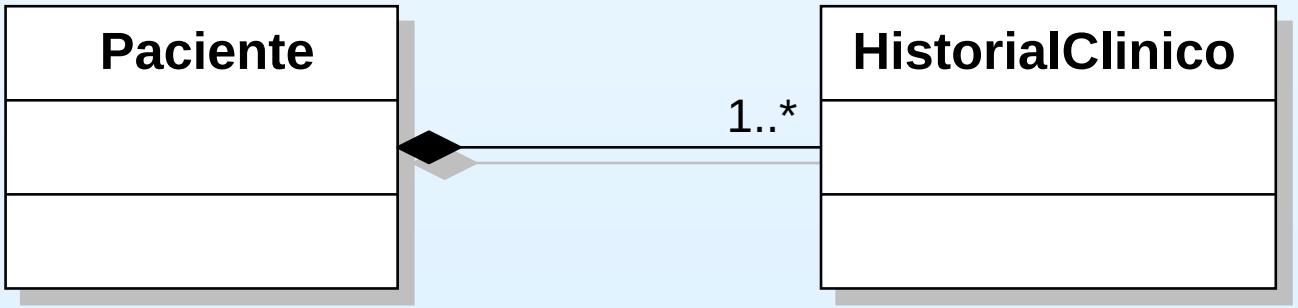
Agregación simple

- No cambia el significado de la navegación entre el todo y sus partes.
- No liga la vida del todo y las partes.
- UML representa la agregación simple mediante una figura en forma de **diamante hueco** sobre el extremo de la línea de la asociación que termina en la clase propietaria.



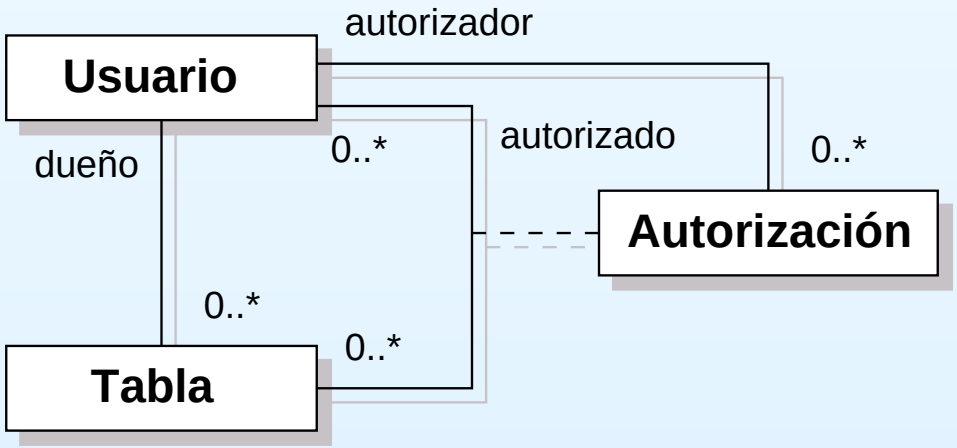
Composición

- Representa una fuerte relación de pertenencia y vidas coincidentes de la parte y el todo.
- Las partes se crean después del objeto al que pertenecen y una vez creadas viven y mueren con él.
- Una parte puede formar parte de sólo un objeto compuesto.
- El objeto compuesto debe gestionar la creación y destrucción de las partes.
- La composición se representa mediante un **diamante sólido** sobre el extremo de la línea de la asociación que termina en la clase propietaria.

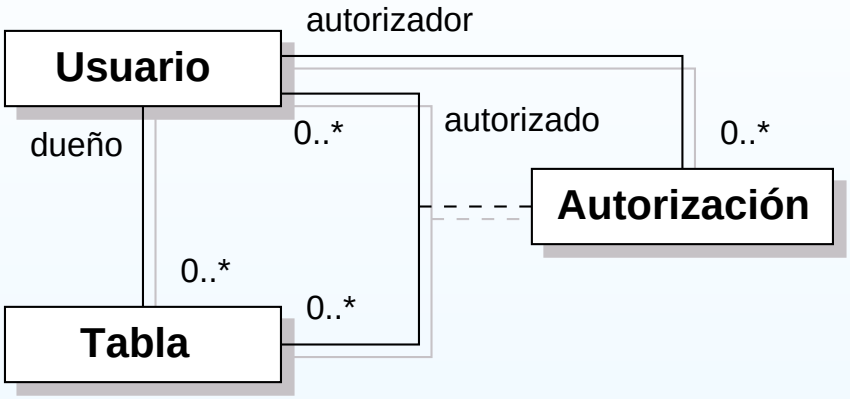


Clases de asociación

- Una *clase de asociación* es una asociación cuyos enlaces pueden participar en asociaciones posteriores.
- Una clase de asociación tiene características de asociación y de clase:
  - Como el enlace de una asociación, las instancias de una clase de asociación obtienen su identidad de las instancias de las clases que las constituyen.
  - Como una clase, pueden participar en asociaciones.



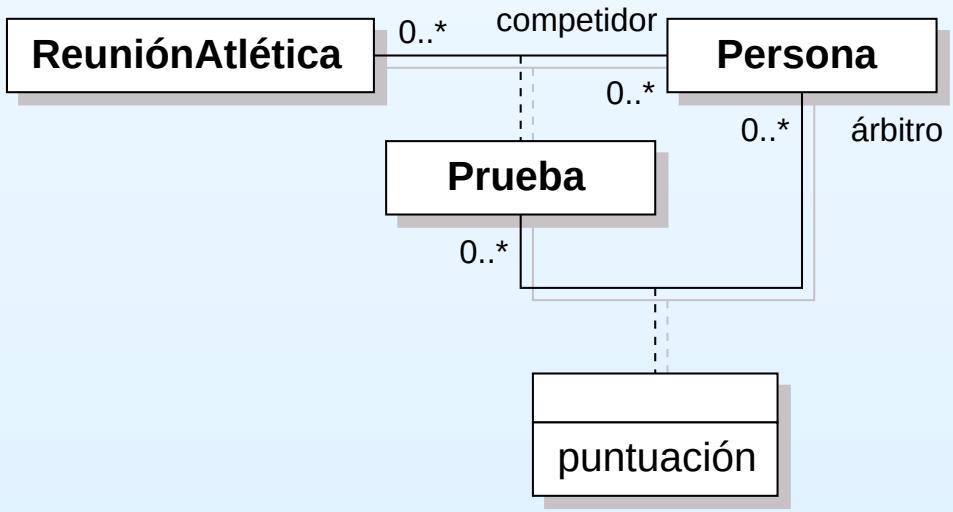
Clases de asociación



- Un usuario puede ser el dueño de múltiples tablas.
- El dueño de una tabla puede autorizar el acceso a dicha tabla a uno o más usuarios.
- Éstos pueden a su vez autorizar a otros usuarios. Por ejemplo:
  - El usuario *A* autoriza el acceso a los usuarios *B* y *C*.
  - El usuario *A* es el autorizador y *B* y *C* los autorizados.
  - *B* puede a su vez autorizar a un usuario *D* a acceder a la tabla.

Clases de asociación

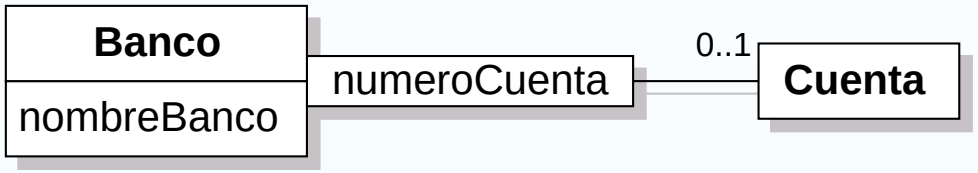
- Las clases de asociación constituyen una faceta importante del modelado UML porque permiten especificar de forma precisa la identidad y el recorrido.
  - En la figura anterior: una tabla y un autorizado implican necesariamente una autorización; un autorizador puede proporcionar varias de dichas autorizaciones.
  - En el *meeting* atlético: **Prueba** es una clase de asociación que describe a una persona que compite en un evento atlético. Un modelo mejorado es:



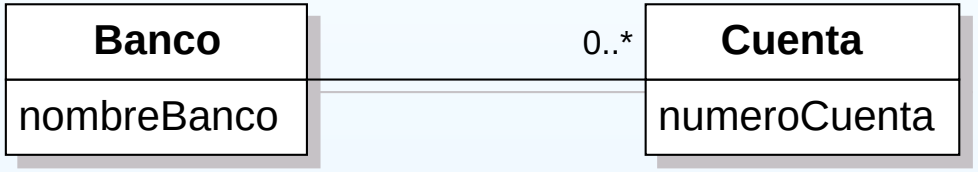
Asociación calificada

- Una *asociación calificada* es una asociación en la cual los objetos en el *rol* de “muchos” se pueden identificar de forma no ambigüa mediante un atributo denominado *calificador*.
- Un calificador actúa como un selector entre los objetos reduciendo la multiplicidad efectiva de “muchos” a “uno”.
- Una asociación calificada con una multiplicidad “uno” o “cero o uno” especifica un camino preciso para encontrar el objeto destino a partir del objeto fuente.

Asociación calificada



Asociación cualificada

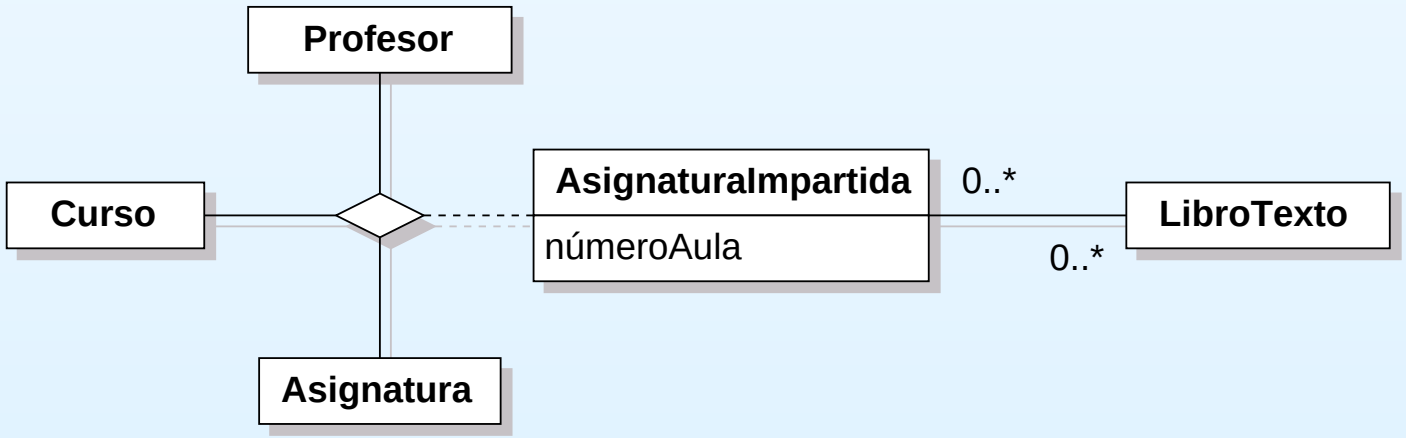


Asociación no cualificada

- Ambos modelos son aceptables, pero el modelo cualificado aporta más información:
  - Añade una restricción en la multiplicidad: la combinación de un banco y un número de cuenta conduce a una única cuenta bancaria.
- El modelo cualificado es más adecuado, ya que un número de cuenta no tiene sentido si no se especifica también el banco.

Asociaciones ternarias

- El *grado de asociación* es el número de *roles* de cada enlace.
  - La gran mayoría de las asociaciones son binarias o binarias-cualificadas.
  - Una *asociación ternaria* es una asociación con tres roles que no pueden representarse mediante asociaciones binarias.
  - Las asociaciones ternarias son poco frecuentes y es muy difícil encontrar asociaciones de grado mayor.



Concepto de generalización

Generalización es la relación entre una clase (la *superclase*) y una o más variaciones de esta clase (las *subclases*).

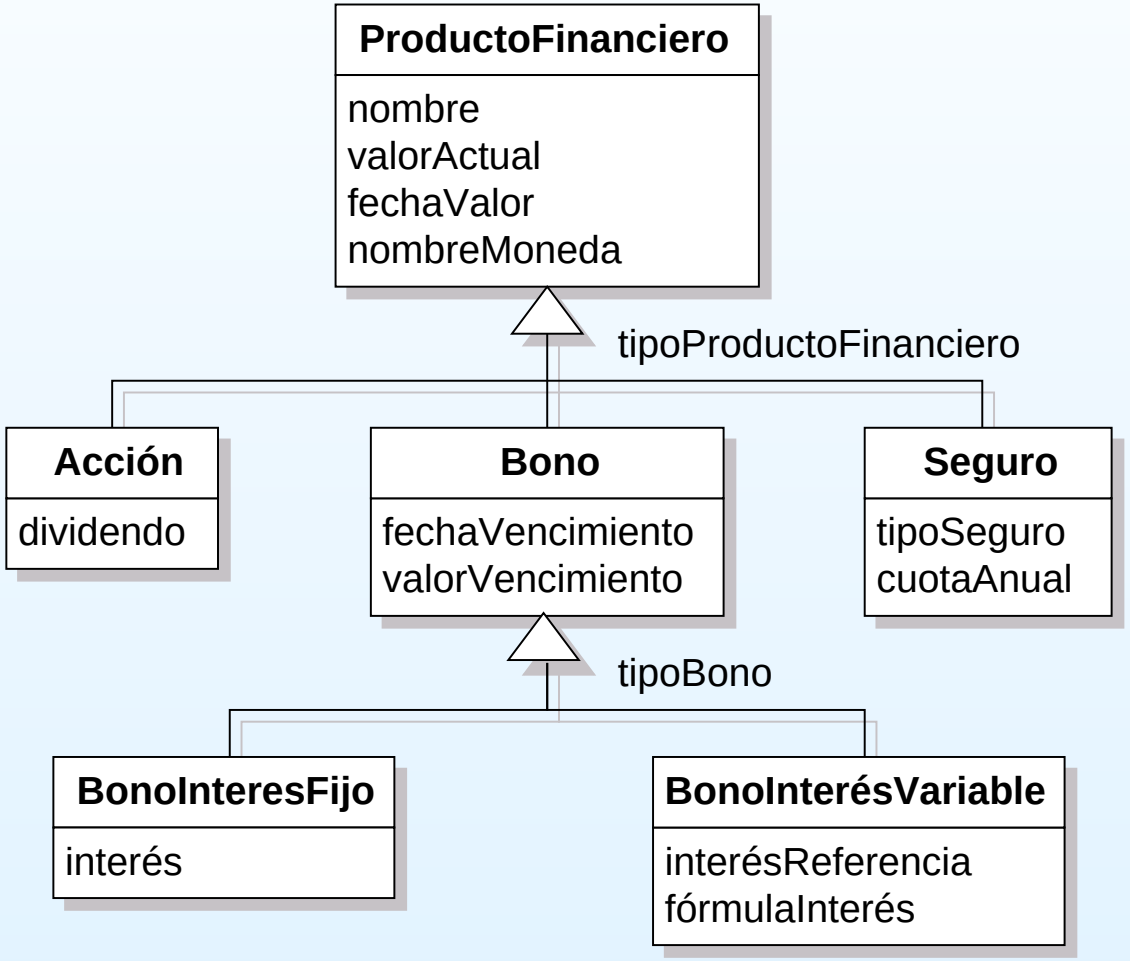
- Organiza las clases de acuerdo con sus similitudes y diferencias.
- Proporciona estructura a la descripción de los objetos.
- La superclase contiene los atributos, operaciones, diagramas de estado y asociaciones comunes.
- La subclase añade atributos, operaciones, diagramas de estado y asociaciones específicos.
- Una instancia de una subclase es también una instancia de todas sus superclases.

Concepto de especialización

La *especialización* proporciona otra prespectiva de la estructura del sistema. Tiene el mismo significado que generalización pero aporta una visión de arriba a abajo.

Generalización

La *generalización simple* organiza las clases en una jerarquía en la que cada subclase tiene una única superclase inmediata.



Generalización

- Todos los productos financieros tienen un nombre y un valor actual que se expresa en una determinada moneda.
- Las acciones tienen un dividendo que es decidido por los directores de la compañía.
- Los bonos tienen fecha y valor de vencimiento además de un tipo de interés bien fijo, bien variable y calculado a partir de un interés de referencia.
- En la clase **Seguro** se tiene en cuenta que existen diferentes tipos de seguros: invalidez, vida, riesgo. . .

Herencia

- La generalización es la relación estructural que permite la existencia del mecanismo de *herencia*.
- Una subclase hereda los atributos, operaciones, diagramas de estado y asociaciones de su superclase.
- Las propiedades heredadas pueden reutilizarse o redefinirse en la subclase.
- La subclase puede definir nuevas propiedades no presentes en la superclase.
- La generalización simple es otro término para *herencia simple*.
- El caso en el que una subclase tiene múltiples superclases inmediatas se denomina *herencia múltiple*.

Ejemplos: Diagrama de clases

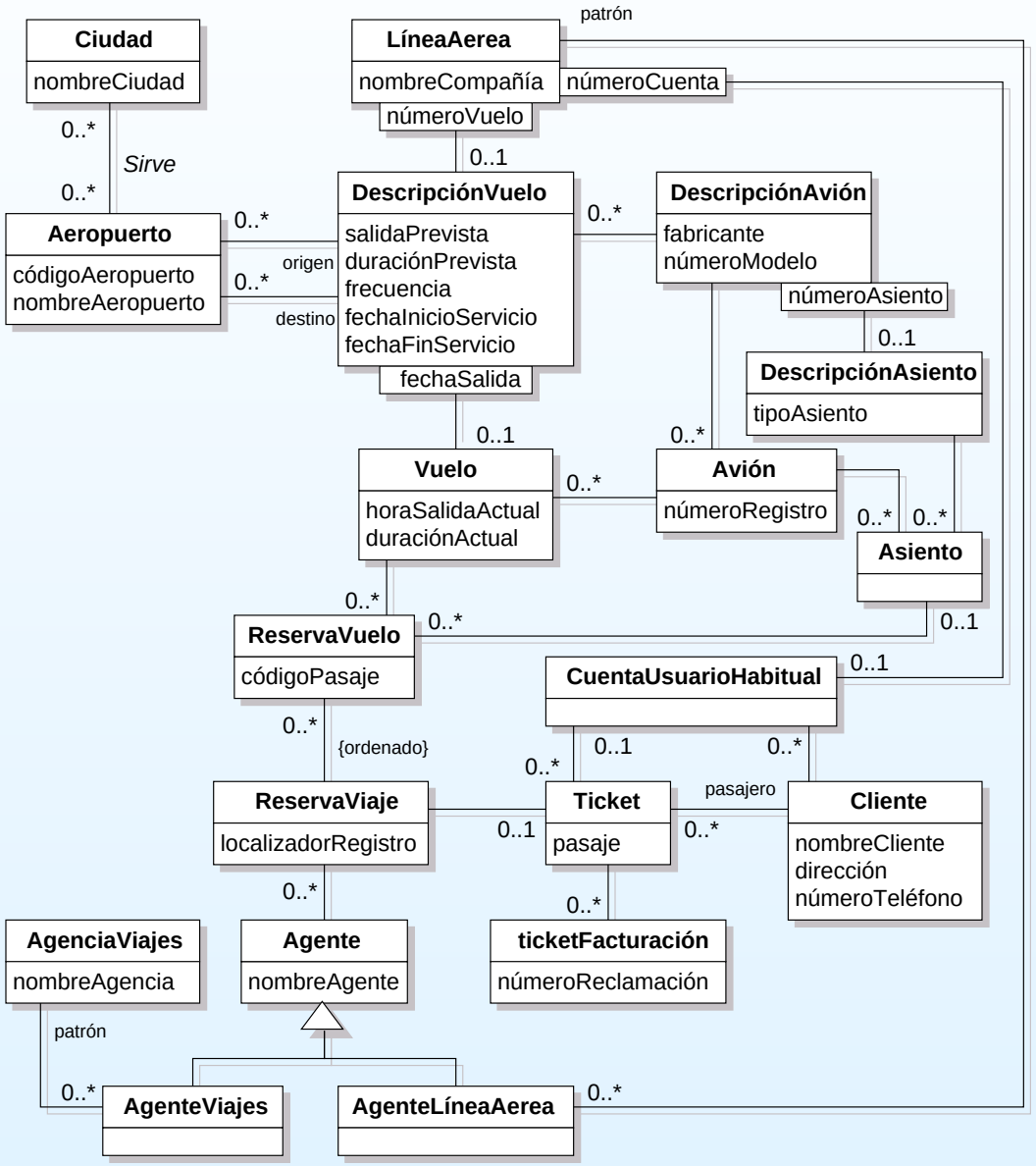


Diagrama de clases . . .

Un *diagrama de clases* es la representación gráfica que describe los objetos y sus relaciones.

1. Un aeropuerto sirve a varias ciudades y una ciudad puede tener varios aeropuertos.
2. Las líneas aéreas ofertan vuelos entre aeropuertos.
3. Descripción de vuelo hace referencia a la información pública que describe un tipo de vuelo entre dos aeropuertos. Un vuelo hace referencia al viaje concreto que lleva a cabo un avión en una fecha concreta.

Diagrama de clases . . .

4. Normalmente, se registran varias reservas para un vuelo, cada una con su código de pasaje. Los códigos de pasaje determinan el precio del pasaje. Los códigos de pasaje reflejan múltiples factores, tales como:
   - El número de días que hace que se compró el pasaje.
   - La posibilidad de devolución.
   - Si el pasajero pasará el fin de semana en la ciudad de destino.
   - El tipo de asiento (*tipoAsiento*).
5. Un pasajero tiene asignado un asiento junto con su reserva de vuelo. Un asiento puede estar relacionado con multitud de reservas de vuelo, pero desde luego, no más de una por vuelo.

Diagrama de clases . . .

6. Una reserva de viaje consiste en una secuencia de reservas de vuelo.
7. La reserva de viaje la realizan los agentes, que pueden trabajar bien para la compañía aérea, bien para una agencia de viajes.
8. Para cada pasajero que hace una reserva de viaje es necesario un ticket y en ocasiones, utilizar además una cuenta de usuario (en el caso de clientes habituales).

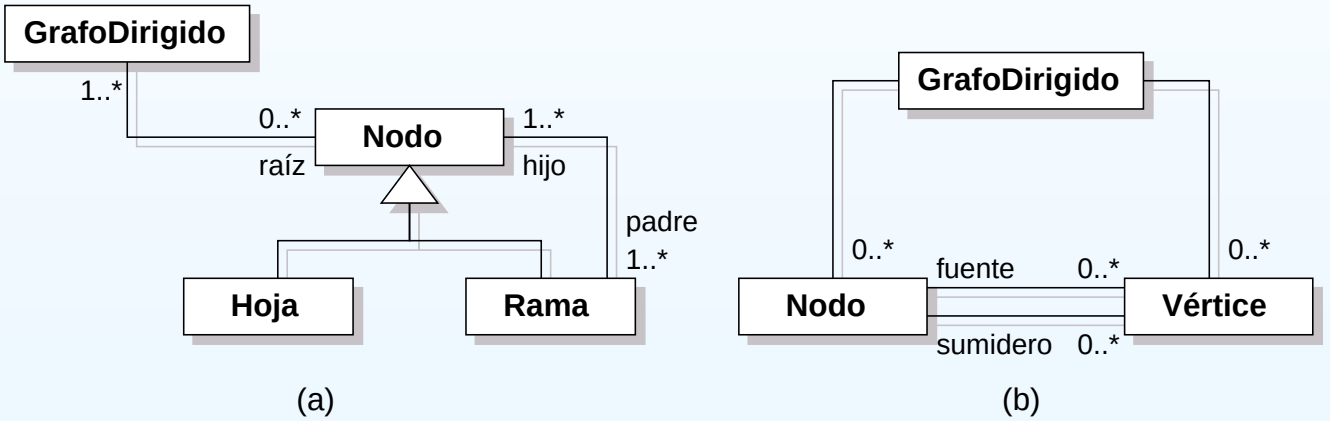
Diagrama de instancias

Un *diagrama de instancias* implica sólo objetos, enlaces y valores.

- Los diagramas de instancias son útiles para depurar y entender los diagramas de clases.
- Los diagramas de instancias pueden ayudar a entender las capacidades y deficiencias de las diferentes representaciones.
- Ejemplo: analicemos los dos diagramas de clases que modelan el comportamiento de un grafo dirigido:

Diagrama de instancias

Un grafo dirigido consiste en un conjunto de nodos y vértices. Cada vértice conecta, mediante una flecha que indica la dirección, dos nodos.



- Modelo (a) describe un gráfo dirigido con al menos un vértice entre nodos; los vértices se representan de forma implícita mediante la asociación padre-hijo entre **Rama** y **Nodo**. Por definición, cada nodo tiene uno o más padres excepto el nodo raíz.
- Modelo (b) es más potente y puede describir cualquier grafo dirigido.

Diagrama de instancias . . .

En el esquema se muestra un grafo dirigido de ejemplo y los diagramas de instancias asociados a cada diagrama de clases.

Grafo dirigido X

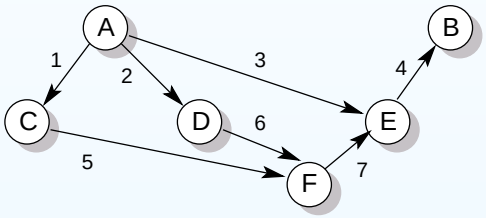


Diagrama de instancias para el modelo (a)

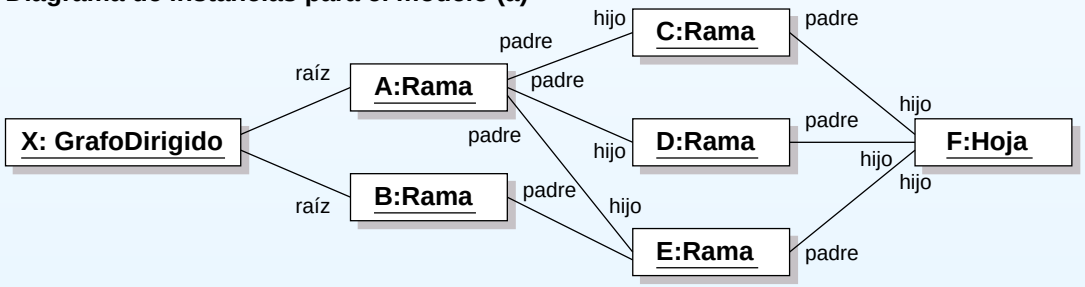


Diagrama de instancias para el modelo (b)

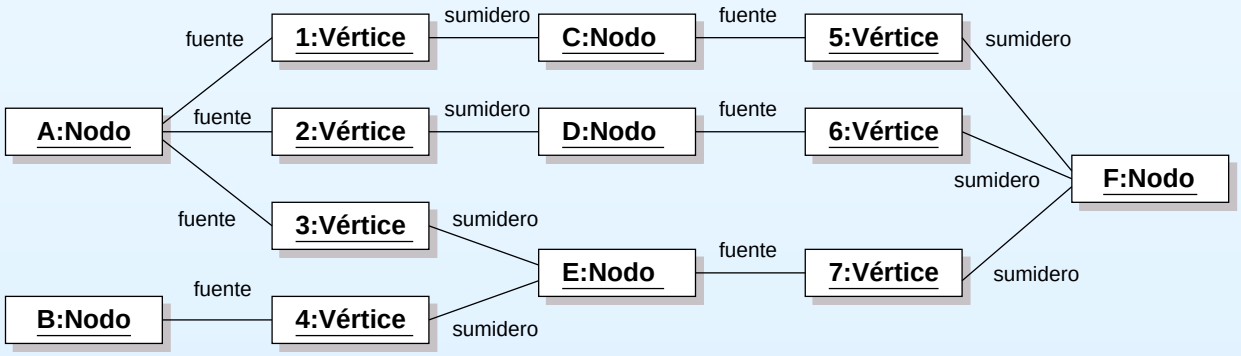


Diagrama de instancias . . .

- Modelo (a): se corresponde directamente con el grafo (nodos que se conectan mediante vértices).
- Modelo (b) es más complejo: nodos y vértices son objetos.